# HTL Saalfelden

## Systemplanung und Projektentwicklung

## Projektdokumentation
## 2024 / 2025

| | |
|---|---|
| **Projektbezeichnung** | MBOT Lore |
| **Projektteam** | Johannes Ellmer, Christos Magos, Timo Winkler |
| **Erstellt am** | 22.01.2025 |
| **Letzte Änderung am** | 11.06.2025 |
| **Status** | fertiggestellt |
| **Aktuelle Version** | 1.0 |

# Inhalt

# 1. Allgemeines / Projektübersicht

## 1.1 Projektbeschreibung

Dieses Projekt konzentriert sich auf die Entwicklung eines webbasierten Frontend zur Steuerung eines MBots. Die Anwendung bietet Funktionen wie manuelles und automatisches Fahren, Routenaufzeichnung und -speicherung, LED-Steuerung über einen RGB-Picker sowie Sitzungswiederherstellung. Das Frontend kommuniziert ausschließlich über eine definierte API-Schnittstelle mit dem Backend, dass die eigentliche Steuerlogik und Hardwareansteuerung übernehmen.

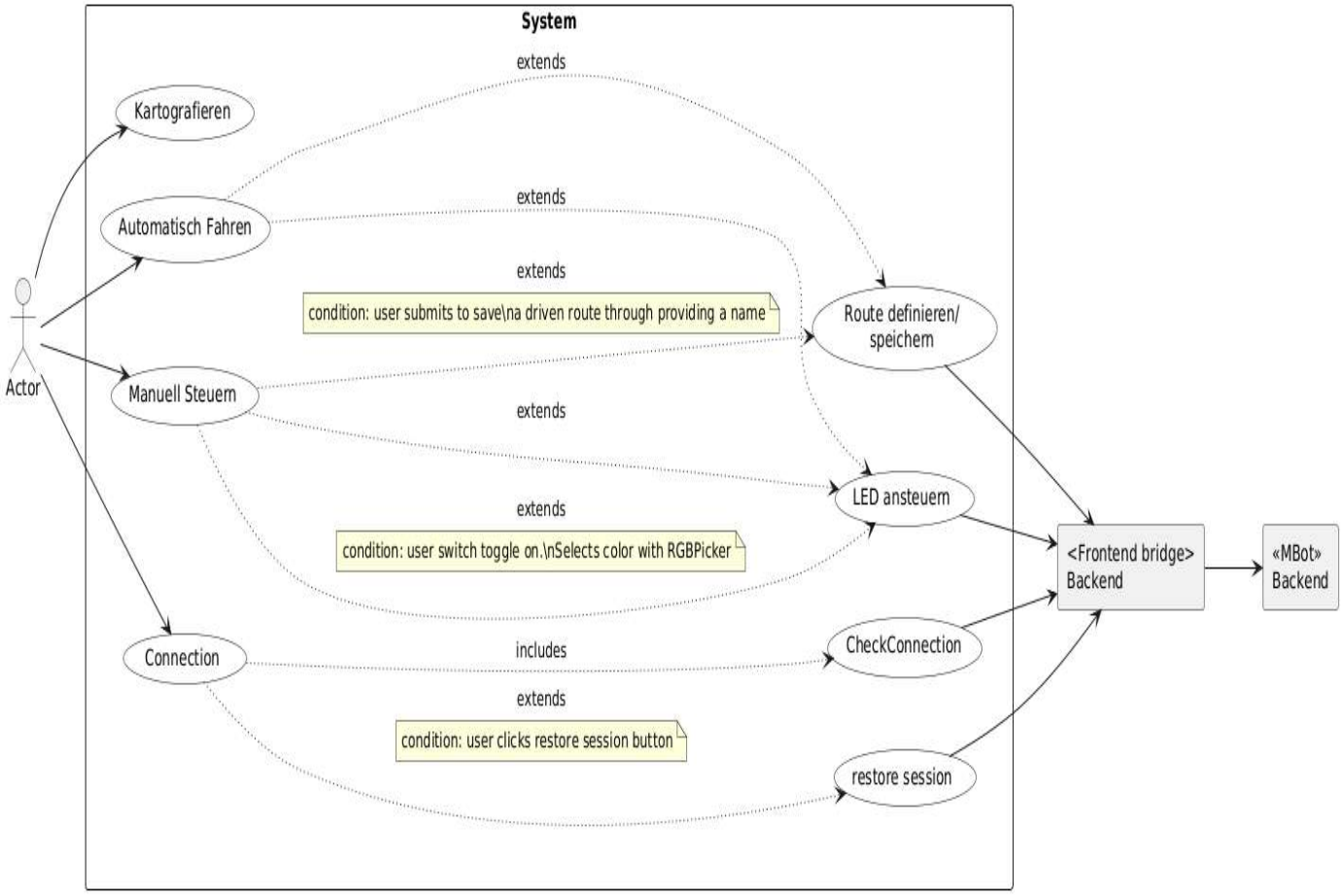
Benutzer können über die Oberfläche Fahrbefehle geben, Routen verwalten und visuelle Feedbacksysteme aktivieren. Bedingte Funktionen wie das Speichern einer Route oder die LED-Steuerung sind kontextabhängig und werden dynamisch im UI freigeschaltet. Das Projekt legt besonderen Wert auf eine klare Trennung zwischen Benutzeroberfläche und Backend-Logik und ermöglicht so eine flexible Erweiterung oder Integration in andere Systeme.

## 1.2 Projektteam und Schnittstellen

| Name | E-Mail | Team |
|---|---|---|
| Ellmer Johannes | Johannes.ellmer@htl-saalfelden.at | MBot Lore Frontend |
| Christos Magos | Christos.magos@htl-saalfelden.at | MBot Lore Frontend |
| Timo Winkler | Timo.winkler@htl-saalfelden.at | MBot Lore Frontend |

# 2. Funktionale Anforderungen

## 2.1 Use Cases




### 2.1.1 Connection:

Der User gibt IP-Adresse und beliebigen Namen des Mbots ein, um sich mit ihm zu verbinden. Hierbei kann der Benutzer die Daten einer vorherigen Session mittels Restore Session abrufen und anschließend eine Verbindung aufstellen. Anschließend wird die Verbindung aufgestellt und überprüft.

### 2.1.2 Restore Session:

Bei der Connection-page kann sich der Benutzer sich vergangene Sessions anzeigen lassen bzw. wieder mit diesen Werten verbinden. Dazu muss der Benutzer nur den „Restore Session“-Button betätigen. Danach öffnet sich eine Liste an Sessions, die angeklickt werden können. Wenn das getan wurde, werden die Eingabefelder automatisch ausgefüllt. Dadurch muss der Benutzer sich nicht die IP-Adresse des Mbots, den er schon einmal verwendet hat, merken.

### 2.1.3 Manuelle Steuerung:

Nach dem der Benutzer sich mit dem Mbot erfolgreich verbunden hatte, kann er „manuelle Steuerung“ auswählen. Dabei muss der User den „drive“-Button drücken. Dadurch wird einerseits das manuelle Steuern aktiviert und andererseits im Hintergrund das Aufzeichnen der Route gestartet. Jetzt kann der Benutzer je nach Belieben die Farbe und die Geschwindigkeit ändern. Außerdem kann der Benutzer den MBOT steuern. Wenn der Benutzer fertig gefahren ist, klickt er auf stop Drive. Dann wird der Fahrmodus beendet und er bekommt die Anfrage zum Speichern der gefahrenen Route. Dieser Route kann so gespeichert werden und bei Bedarf später z.B. beim automatischen Abfahren von Routen wiederverwendet werden.

### 2.1.4 Automatisches Fahren

Der Benutzer kann im automatischen Modus, eine gefahrene oder gespeicherte Route laden. Die gespeicherten Routen können per Name über ein Dropdown ausgewählt werden. Nach dem erfolgreichen Speichern der Route kann diese ausgewählt werden und nach dem Betätigen des Start-Buttons beginnt der MBot die Route abzufahren.

### 2.1.5 Route definieren

Um eine Route zu definieren kann der User entweder eine gefahrene Route speichern oder beim automatischen Fahren ein Fenster öffnen, in dem verschiedene Checkpoints als Liste für eine Route gespeichert werden. Dabei können Dauer, Geschwindigkeit, Farbe und Richtung innerhalb dieses Abschnittes gespeichert werden. Diese Route kann später ausgewählt und abgefahren werden.

### 2.1.6 Farbe ändern:

Um die Farbe zu ändern, muss zuerst die Farbe eingeschalten werden. Dazu muss der Benutzer das „toggle“ der Farbe auf on stellen. Danach soll er auf das Quadrat daneben drücken. Darin wird die aktuell ausgewählte Farbe angezeigt. Bem Click darauf, öffnet sich der ColorPicker, in dem eine beliebige Farbe des RGB-Schemas ausgewählt werden kann.

### 2.1.7 Kartografieren:

Beim Kartografieren wählt der User zunächst das MapControl-Fenster aus und kommt anschließend zur MapControl-Page. Darin befinden sich verschiedene Buttons, mit denen das Kartografieren gestartet werden kann. Beim Betätigen startet der Mbot und beim erneuten Klicken wird das Kartografieren beendet. Es öffnet sich nun ein weiteres Fenster, das MapPopup. Dieses zeigt die erfassten Punkte Hindernisse und hilft beim Kartografieren. Die Daten dafür kommen vom Backend, darin steht der Winkel und die Richtung.

# 3. Nichtfunktionale Anforderungen

**1. Benutzerfreundlichkeit (Usability)**

- Die Benutzeroberfläche soll intuitiv und selbsterklärend sein.
- Die Steuerung (Joystick, Routenplanung, Kartografieren) soll einfach bedienbar sein.
- Konsistente Darstellung auf Desktop- und Mobilgeräten (responsives Design).

2. **Performance**

- Echtzeit-Darstellung der Sensordaten (geringe Latenz bei der Aktualisierung).
- Flüssige Visualisierung der Roboterbewegungen und Karten.
- Schnelle Reaktion auf Benutzereingaben (z. B. Joystick-Steuerung).

**3. Kompatibilität & Plattformunabhängigkeit**

- Unterstützung für gängige Browser
- Mobile Kompatibilität

**4. Zuverlässigkeit**

- Stabile Verbindung zum Backend
- Fehlertoleranz bei falschen Benutzereingaben
- Korrekte Anzeige der Roboter-IP und des Verbindungsstatus.

# 4. Projektplanung

## 4.1 Variantenbildung

| Kriterium | Variante A (Minimal) | Variante B (Standard) | Variante C (Erweitert) |
|---|---|---|---|
| Frontend-Technologie | Nur Desktop-Browser (React oder Blazor Web) | Responsives Web-Frontend für Desktop & Mobile (z. B. Blazor oder Angular mit PWA) | Native Apps (Flutter oder React Native) zusätzlich zum responsiven Web-Frontend |
| Bedienmodi | Nur manuelle Steuerung (Joystick) | Manuell + Routensteuerung | Manuell + Routen + Kartographierung + Replay |
| Visualisierung der Karte | Textuelle Ausgabe der Koordinaten | 2D-Karte mit einfacher Darstellung (Canvas oder SVG) | Interaktive Karte mit Zoom, Hindernisfarben, Pfadanimationen etc. |
| Kommunikationsart | Nur lokale IP manuell eingeben | Automatische Discovery im LAN (z. B. UDP-Broadcast) | Kombination mit MQTT oder WebSockets für Live-Daten |
| Sensorintegration | Nur Bewegung und Abstand | Abstand, Licht, Lagesensor, LEDs | Alle Sensoren inkl. konfigurierbarer LED-Steuerung über das Frontend |

## 4.2 Allgemeine Planungsinformationen

**1. Ziel des Frontends**
Das Frontend dient als zentrale Benutzeroberfläche zur:

- Steuerung des MBot2 (manuell, vordefinierte Route, Aufzeichnung, Kartographierung),
- Anzeige aller relevanten Sensordaten (Abstand, Licht, Lage, LEDs, Geschwindigkeit, Betriebsdauer),
- Interaktion mit dem Benutzer (Befehle senden, Daten anzeigen, Feedback geben),
- Visualisierung von Routen, Karten und Hindernissen.

---

## 4.3 Projektumfeldanalyse

### Technisches Umfeld

| Aspekt | Beschreibung |
|---|---|
| **Hardware** | MBot2-Roboter mit CyberPi-Controller, diverse Sensoren, WLAN-Fähigkeit |
| **Software** | MicroPython, mBlock für Testzwecke, Webtechnologien (z. B. Blazor oder Angular) |
| **Plattformkompatibilität** | Anwendung muss auf PC und Mobilgerät lauffähig sein (responsive Web-App) |
| **Schnittstellen/API** | Kommunikation zwischen Frontend und Backend über definierte REST-API |

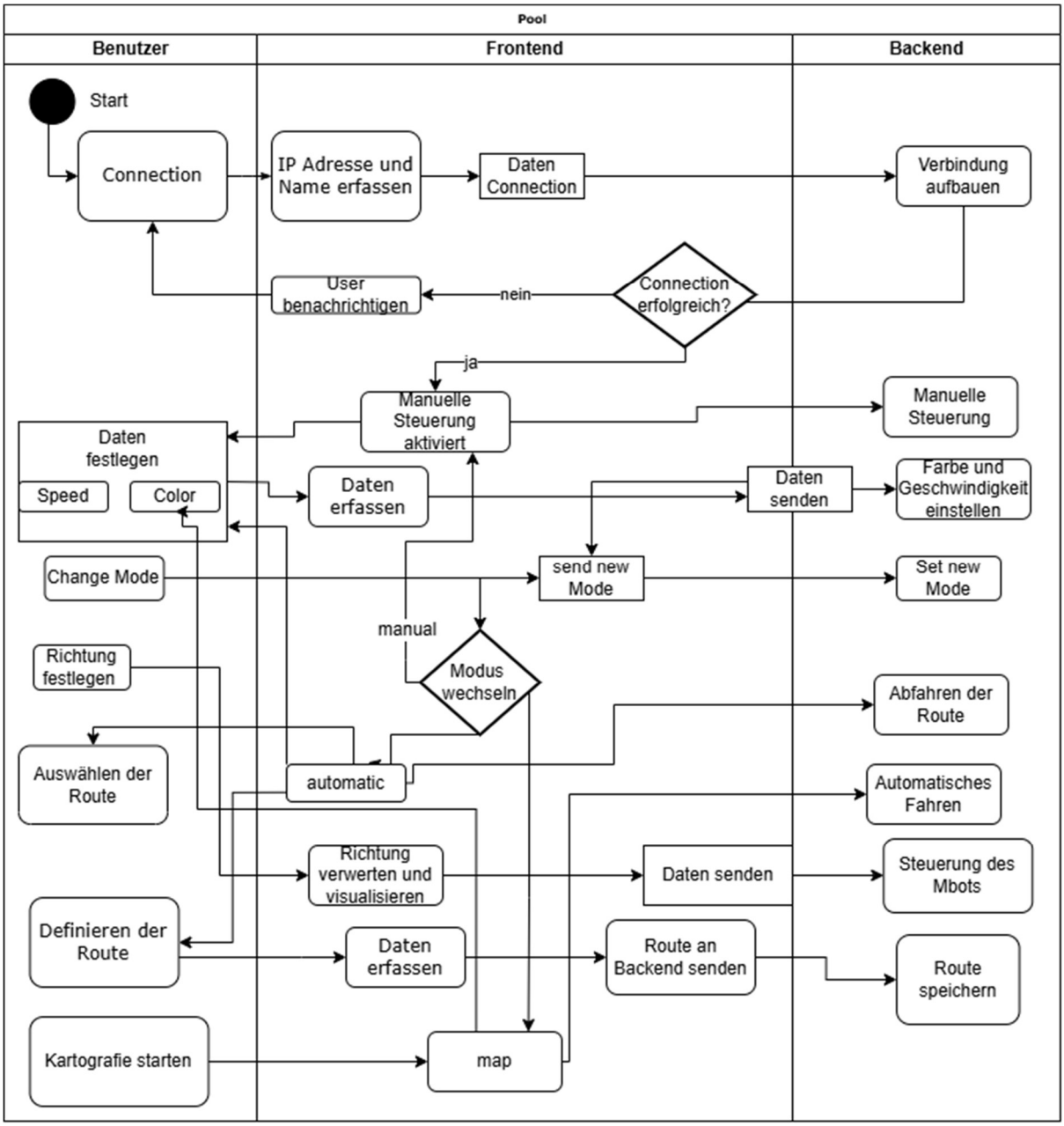
### Soziales Umfeld

| Aspekt | Beschreibung |
|---|---|
| **Zielgruppe** | Schüler:innen können den Roboter programmieren und steuern |
| **Teamdynamik** | Zusammenarbeit und klare Aufgabenverteilung zwischen Frontend und backend |
| **Kommunikation** | Regelmäßige Meetings (Stand-ups, Sprint-Reviews), Online-Kommunikationstools |
| **Wissenstransfer** | Dokumentation für zukünftige Schülergenerationen oder andere Teams |

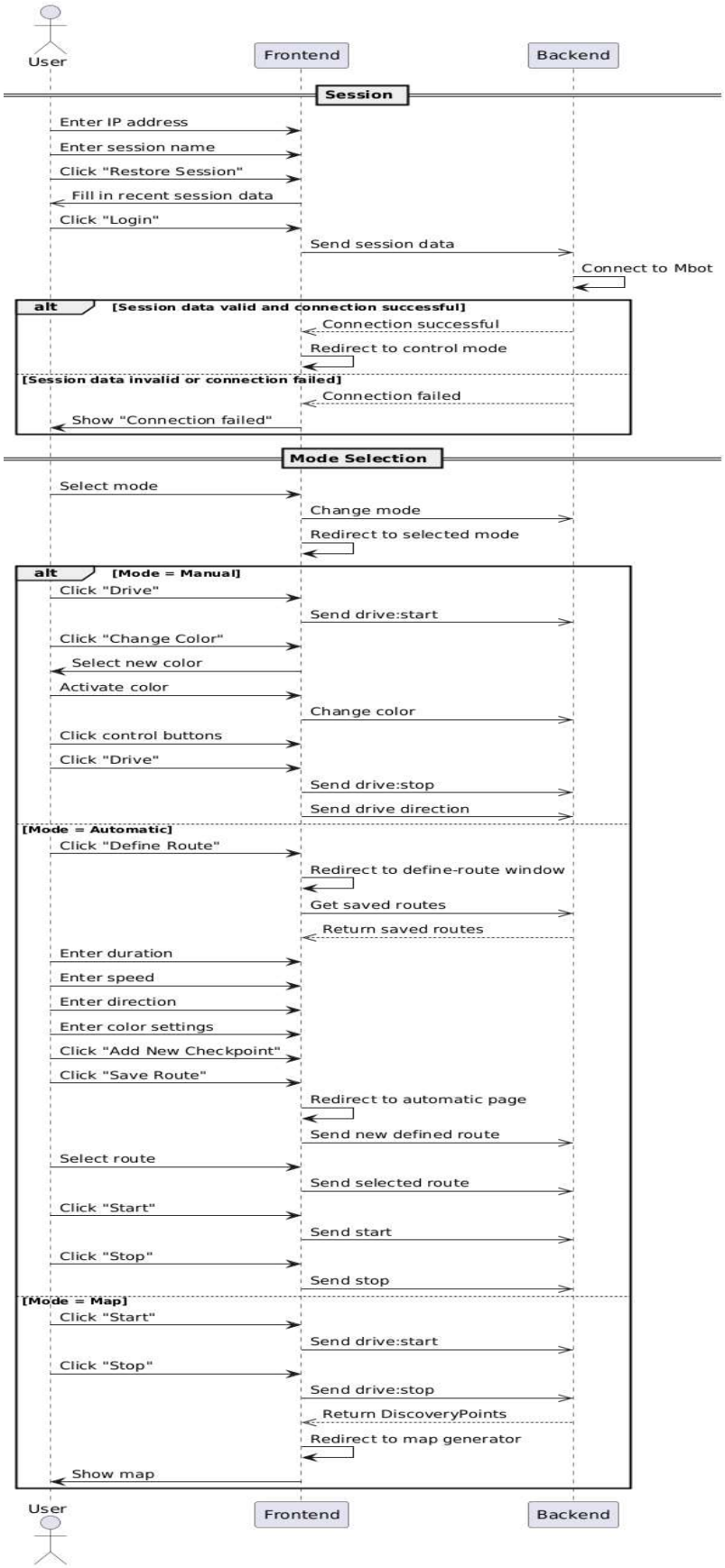
# 5. Softwarearchitektur

Die Diagramme wurden mittels PlantUML, VisualParadigm und Draw.io erstellt.

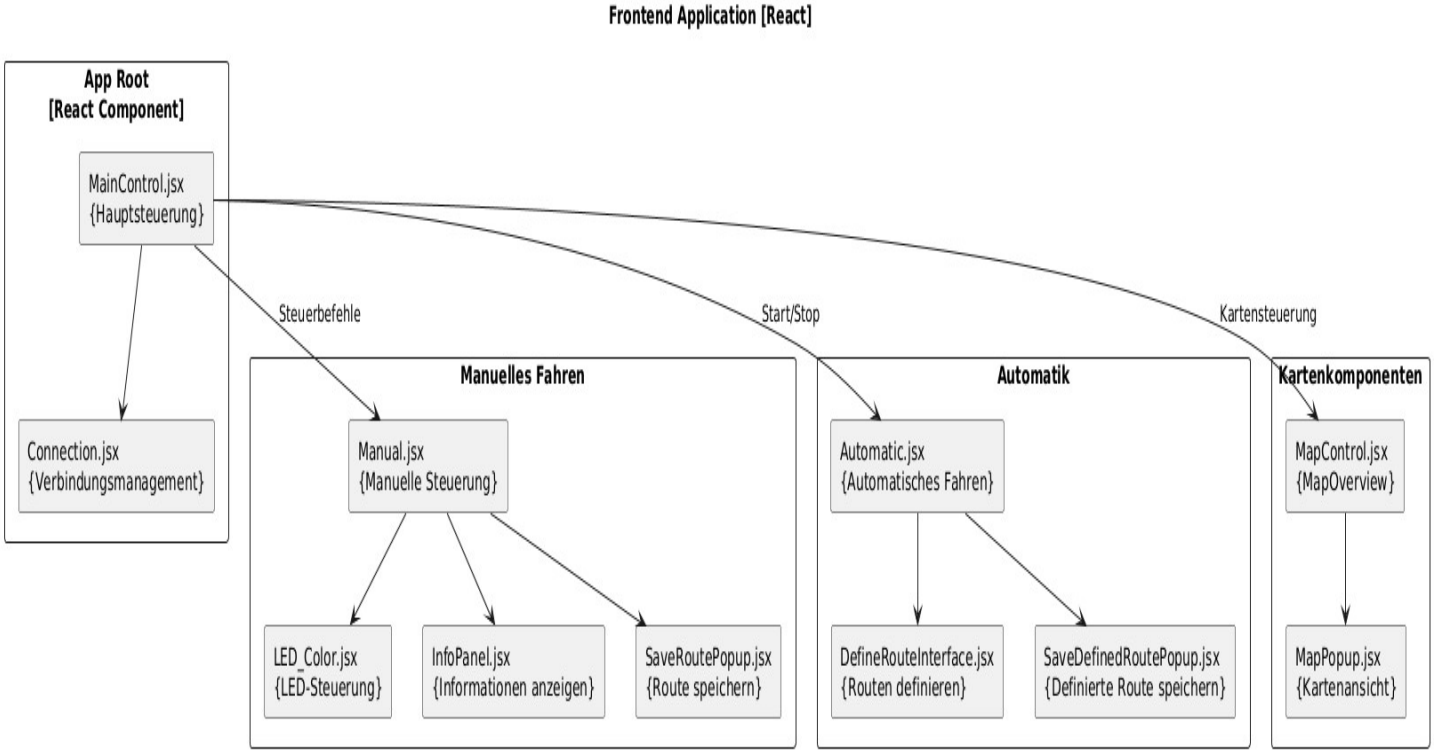
## 5.1 Aktivitätsdiagramm



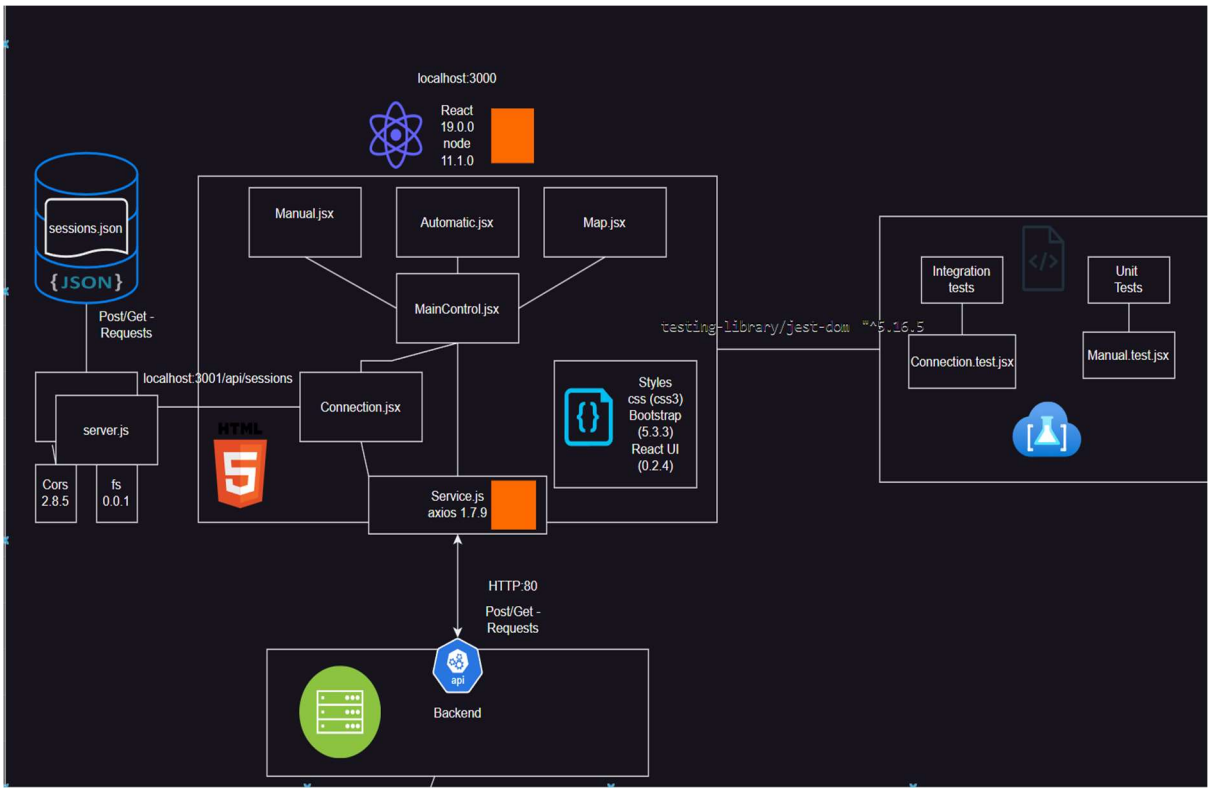
## 5.2 Sequenzdiagramm



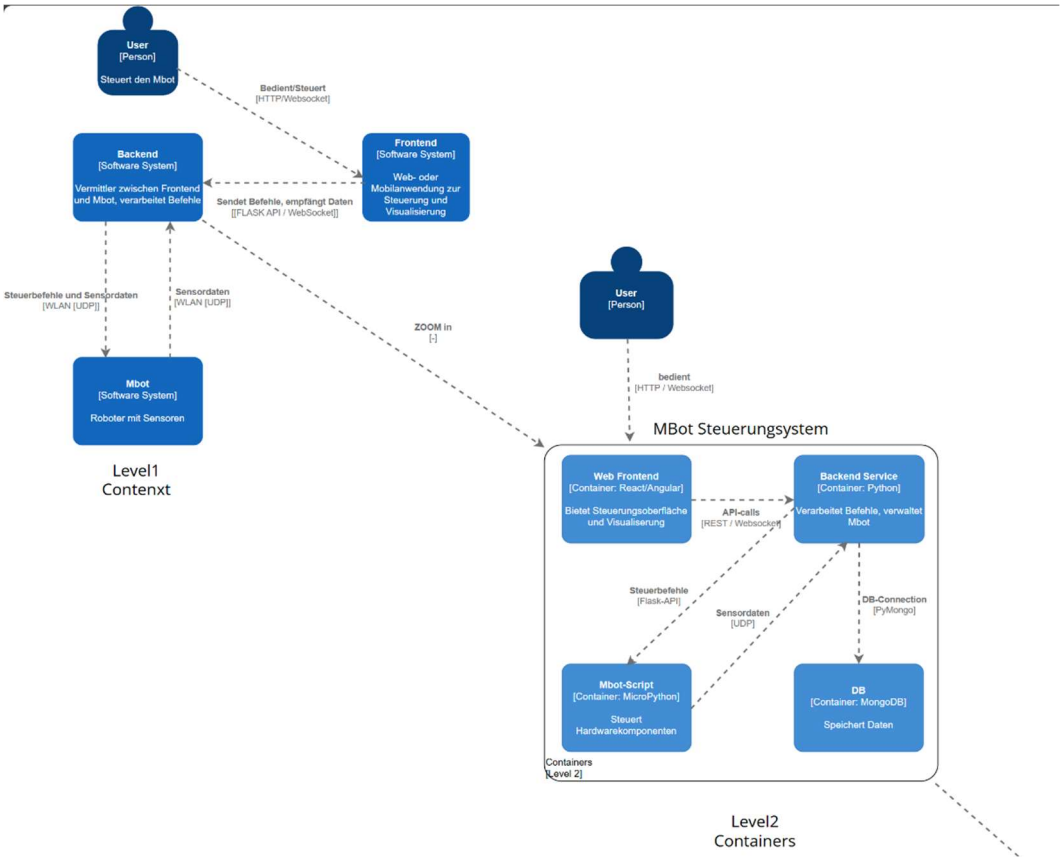
## 5.3 Komponentendiagramm

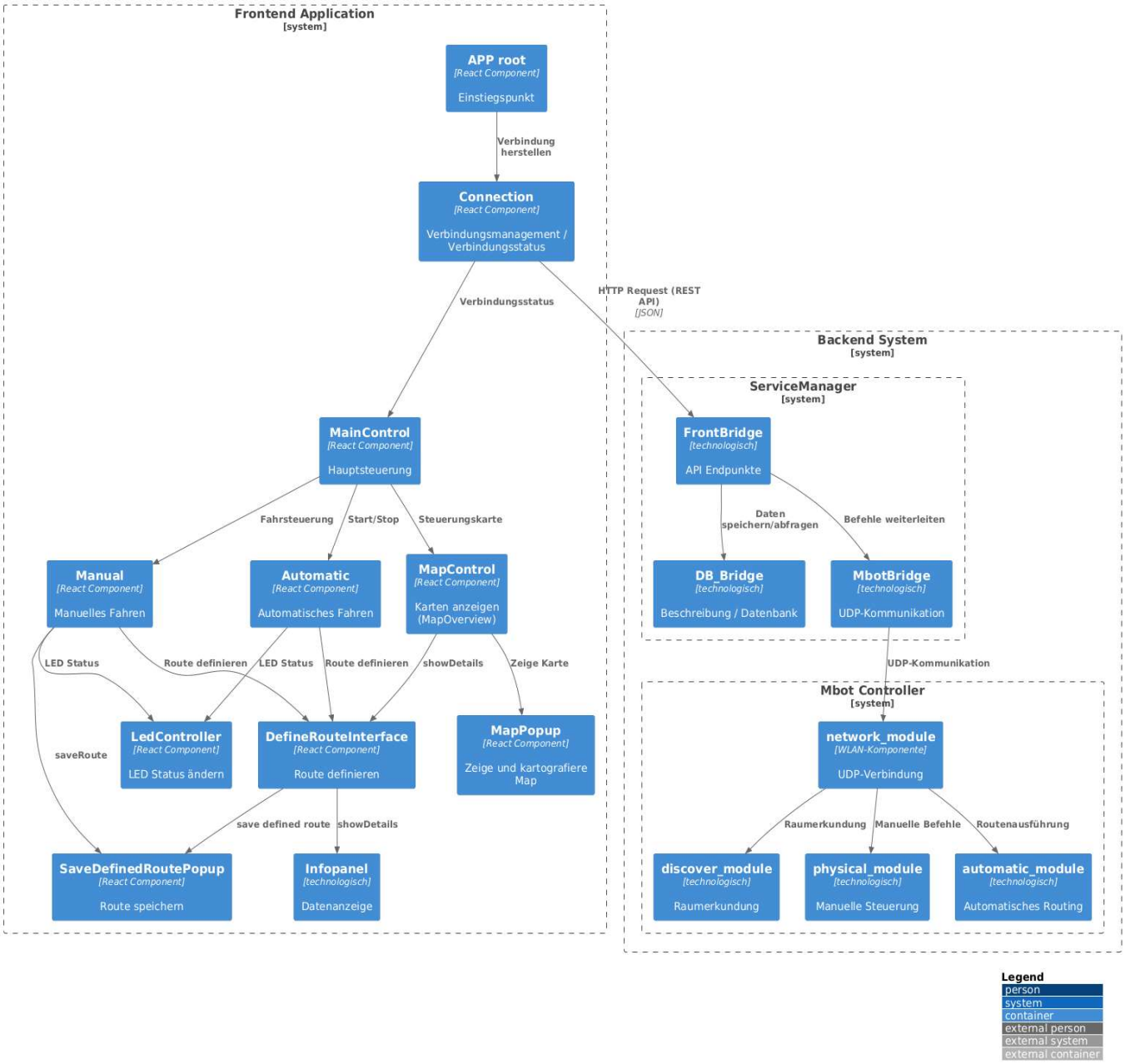


## 5.4 Verteilungsdiagramme

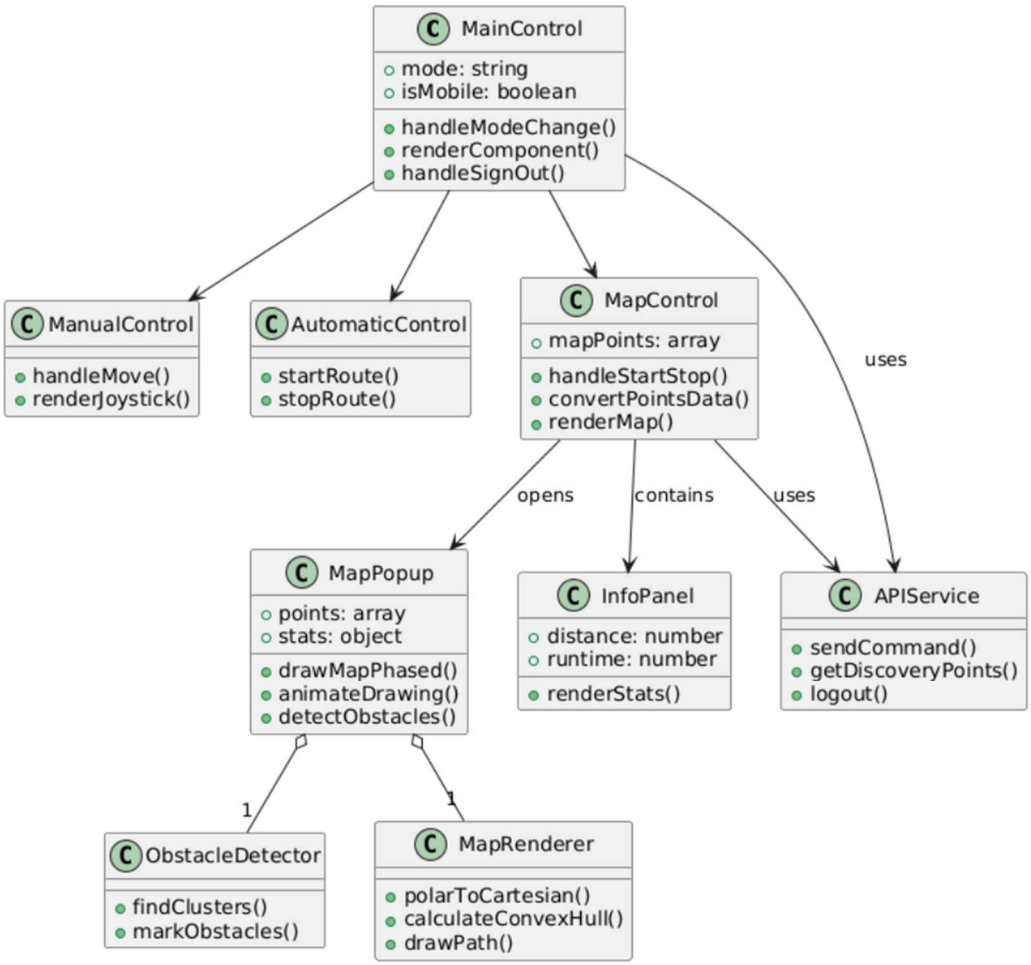



## 5.5 C4 Diagramm




Frontend Application
[system]
APP root
[React Component]
Einstiegspunkt
Verbindung herstellen
Connection
[React Component]
Verbindungsmanagement / Verbindungsstatus
Verbindungsstatus
HTTP Request (REST API)
[JSON]
MainControl
[React Component]
Hauptsteuerung
Fahrsteuerung
Start/Stop
Steuerungskarte
Manual
[React Component]
Manuelles Fahren
Automatic
[React Component]
Automatisches Fahren
MapControl
[React Component]
Karten anzeigen (MapOverview)
LED Status
Route definieren
LED Status
Route definieren
showDetails
Zeige Karte
saveRoute
LedController
[React Component]
LED Status ändern
DefineRouteInterface
[React Component]
Route definieren
MapPopup
[React Component]
Zeige und kartografiere Map
save defined route
showDetails
SaveDefinedRoutePopup
[React Component]
Route speichern
Infopanel
[technologisch]
Datenanzeige
Backend System
[system]
ServiceManager
[system]
FrontBridge
[technologisch]
API Endpunkte
Daten speichern/abfragen
Befehle weiterleiten
DB_Bridge
[technologisch]
Beschreibung / Datenbank
MbotBridge
[technologisch]
UDP-Kommunikation
UDP-Kommunikation
Mbot Controller
[system]
network_module
[WLAN-Komponente]
UDP-Verbindung
Raumerkundung
Manuelle Befehle
Routenausführung
discover_module
[technologisch]
Raumerkundung
physical_module
[technologisch]
Manuelle Steuerung
automatic_module
[technologisch]
Automatisches Routing
Legend
person
system
container
external person
external system
external container

### 5.5.1 Klassendiagramm:



# Softwarekomponenten / Programme

## 5.5.2 SW Programme

- **Visual Studio Code (1.90)** - Hauptentwicklungsumgebung für die Entwicklung mit React und JavaScript/TypeScript.
- **PlantUML (1.2025.6)** - Erstellung von technischen Diagrammen (z. B. Klassendiagramme, Komponenten-, Sequenz- und Architekturdiagramme).
- **draw.io / diagrams.net (22.1.8)** - Erstellung von technischen Diagrammen (z. B. Klassendiagramme, Komponenten-, Sequenz- und Architekturdiagramme).
- **Visual Paradigm (17.2)** - Erstellung von technischen Diagrammen (z. B. Klassendiagramme, Komponenten-, Sequenz- und Architekturdiagramme).
- **Microsoft Word (2405 / 16.0.17628.20000)** - Textverarbeitung und Projektdokumentation (z. B. Pflichtenheft, Abschlussbericht).
- **Git (2.45.1) / GitHub Desktop (3.3.9)** - Versionsverwaltung, Quellcode-Management und Teamarbeit.
- **Google Chrome (125.0.6422.142), Firefox (128.0), Edge (125.0.2535.92)** - Testen und Debuggen der Anwendung im lokalen und späteren Produktivumfeld.
- **PyCharm (2024.3.2):** Für Burn-Down-Chart und Velocity Chart

## 5.5.2 SW Komponenten

- **React.js (v18.2.0)** - JavaScript-Framework zur Erstellung der Benutzeroberfläche in Komponentenarchitektur.
- **React Router DOM (v6.23.0)** - Routing-Library für Navigation und Seitenstruktur innerhalb der React-Anwendung.
- **Vite (v5.2.0)** - Modernes Build-Tool für Frontend-Projekte mit schneller Entwicklungsumgebung und Hot-Reload.
- **React Color (2**.19.3): Color Picker
- Cors (2.8.5), Web Vitals (4.2.4), Ui (0.2.4)
- JavaScript mit statischer Typisierung zur Verbesserung der Codequalität.
- **Axios (v1.6.8)** - HTTP-Client zur Kommunikation mit Backend-APIs (z. B. Datenabruf, POST-Requests). (API-Service)
- **ESLint (v8.56.0)** - Linter zur Analyse des Quellcodes und Einhaltung von Stilregeln.
- Node.js (v20.12.0) - JavaScript-Laufzeitumgebung zur Ausführung von Entwicklungs- und Buildprozessen.
- **npm (v10.5.0)** - Paketverwaltung für Node.js zur Installation und Verwaltung von Bibliotheken.

# 6. Projektdurchführung

## 6.1 Sprint 1

### 6.1.1 Sprintplanung

Dauer: 29.01.2025 – 18.02.2025

Ausgewählte User Stories:

- Mockup:

  Erstellung und Planung des Designs sowie Umsetzungsplanung der Anforderungen.

  ➔ 14 Story Points

- Aufstellung der Verbindung zum Backend über API-Zugriff:

  Das Frontend sollte über diese API mit dem Backend kommunizieren. Dabei muss das Frontend Daten zur Visualisierung aus der API holen können und aber auch Befehle hinschicken, wie der MBOT fahren sollte.

  ➔ 10 Story Points

- Erstellung der Connection Funktionalität und UI:
  Der User sollte sich mit einem MBot anhand der IP-Adresse verbinden und ihm hier einen Namen geben können. Außerdem sollte der USER eine vorherige Sitzung wiederherstellen können.

  ➔ 20 Story Points

- Steuerung UI:
  Der User soll mit dem System über eine GUI interagieren können. Dabei muss das Fenster für manuelle Steuerung und automatische Steuerung erstellt werden.

  ➔ 27 Story Points

Anzahl Story Points: 71

Ausgewählte Punkte aus der Impediment Liste:

- Missverständnisse und Streitigkeiten beim Design – Farben verbessern
- Kommunikation

### 6.1.2 Sprint Demo

In diesem Sprint wurden alle User Stories wie geplant umgesetzt. Außerdem wurde das geplante gut und nahezuproblemlos umgesetzt. Alle Fehler konnten entfernt und korrigiert.

### 6.1.3 Sprint Retrospektive

Besonders gut verlief die Kommunikation mit dem Backend-Team, mit dem ständig eng zusammengearbeitet wurde. Wir sind außerdem mit dem Aufbau der Connection und Control Pages sehr zufrieden und haben auch intern eine gute Kommunikationsbasis genossen.

Verbesserungsbedarf gibt es bei der Aufgabenverteilung und bessere Kommunikation beim Design, bei dem es noch Unstimmigkeiten gibt.

Auflistung der Impediment Taskliste:

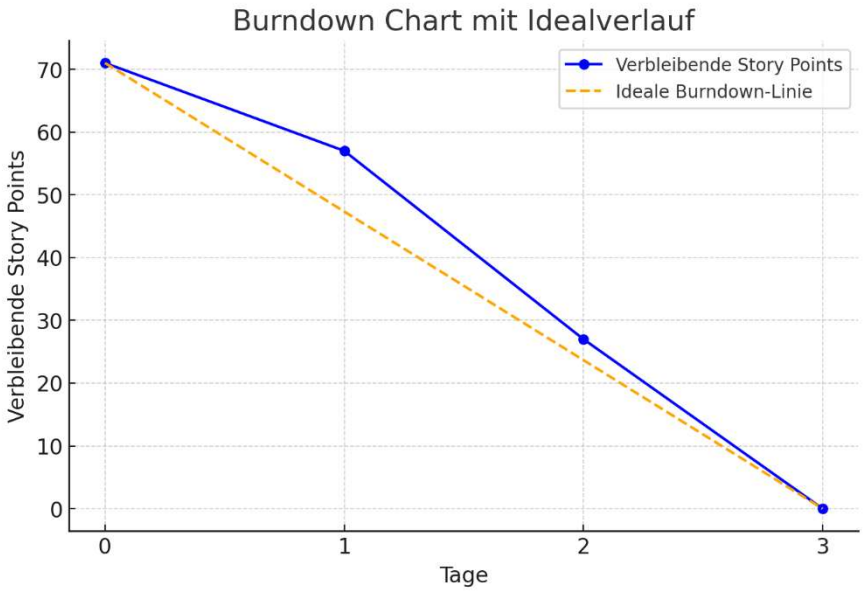
- Missverständnisse und Streitigkeiten beim Design – Farben verbessern

### 6.1.4 Sprint Zusammenfassung

1. Zuerst haben wir uns um das Design gekümmert.
2. Wir haben die Login Page mit Funktionalität eingebaut
3. Wir haben die API-Schnittstelle erstellt.
4. Aufbau und Design der Steuerung Pages

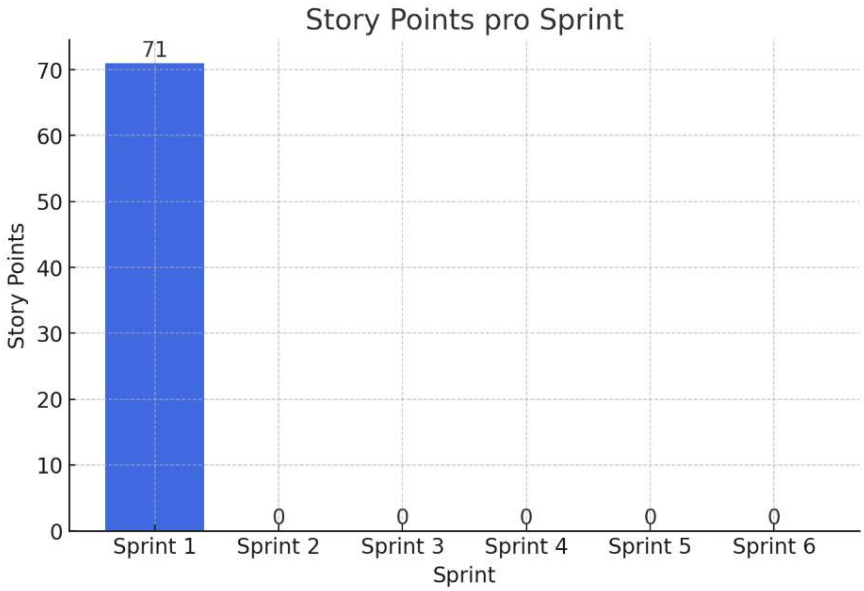
**Während des 1. Sprints hat es keine weiteren Änderungen am Product Backlog gegeben.**

**Burndownchart:**



**Sprint Velocity**

In diesem Sprint hatten wir eine Velocity von 71. Mit einer solchen Geschwindigkeit wäre man in 3-4 Sprints fertig, also ca. 1 Sprint vor dem geplanten Ende. Da es noch keine Vorherigen Sprints lassen sich aber noch keine konkreten Vorhersagungen tätigen.



# 6.2 Sprint 2

## 6.2.1 Sprintplanung

Dauer: 26.02.2025 – 18.03.2025

Ausgewählte User Stories:

- Mockup Handy:

  Erstellung und Planung des Designs sowie Umsetzungsplanung der Anforderungen für die mobile Ansicht.

  ➔ 15 Story Points

- Implementierung der manuellen Steuerung:

  Das Frontend sollte über die API mit dem Backend kommunizieren und dabei Befehle zum Fahren, aber auch für Farbe und Geschwindigkeit mitgeben können. Dafür muss das Frontend die Daten richtig verarbeiten und an das Backend schicken.

  ➔ 20 Story Points

- Abgefahrene Route aufzeichnen und schicken:
  Der User kann bei der manuellen Steuerung automatisch seine Route aufzeichnen lassen. Am Ende wird er gefragt, ob er diese Route speichern will. Diese wird dann an das Backend zum Speichern in der Datenbank geschickt. ob er eine hat sollte sich mit einem MBot anhand der IP-Adresse verbinden und ihm hier einen Namen geben können. Außerdem sollte der USER eine vorherige Sitzung wiederherstellen können.

  ➔ 35 Story Points

  ➔ Datenvisualisierung:
  Kommunikation mit dem Backend um Detaildaten wie die Batterieladung zu holen. Geschwindigkeit etc. werden berechnet und visualisiert.

  ➔ 15 Story Points

Anzahl Story Points: 85

Ausgewählte Punkte aus der Impediment Liste:

- Verzögerung bei Datenvisualisierung
- Kommunikation mit Backend über API → Probleme bei Steuerung (Verzögerung)

### 6.2.2 Sprint Demo

In diesem Sprint gab es leider eine Verzögerung bei der Steuerung, da immer wieder mehrmals dieselben Befehle an das Backend geschickt wurden. Das kostete dem Team viel Zeit und Energie. Schlussendlich konnten aber diese Probleme behoben werden. Die Speicherung der abgefahrenen Route wurde ohne Probleme umgesetzt. Durch die Verzögerung und mangelnde Daten vom Backend für die Visualisierung, konnte die Datenvisualisierung leider nicht bis zum Ende des Sprints fertiggestellt werden und wurde wieder im Sprint Backlog aufgenommen. Insgesamt konnten wir unsere Velocity von ~ 70, trotz Verzögerung halten.

### 6.2.3 Sprint Retrospektive

Besonders gut verlief die Kommunikation mit dem Backend, mit dem wir ständig im Austausch waren. In diesem Sprint wurde das manuelle Fahren vollständig implementiert und begonnen eine Route zu definieren, abzufahren und zu speichern. Des Weiteren haben wir im Team gute Zusammenarbeit genossen, so haben wir oft gemeinsam Probleme und Fehler gelöst. Am Ende haben wir eine ähnliche Velocity wie im Sprint davor, also 70 erreicht. Leider konnte die Datenvisualisierung nicht vollständig umgesetzt werden, da einerseits Daten vom Backend fehlten und allgemein die Visualisierung nicht abgeschlossen wurde. Möglicherweise wurde sich bei der Planung auch überschätzt und es wurde deshalb nicht abgeschlossen.

Verbesserungsbedarf gibt es bei der Aufgabenverteilung und bei dem Vorausplanen.

Auflistung der Impediment Taskliste:

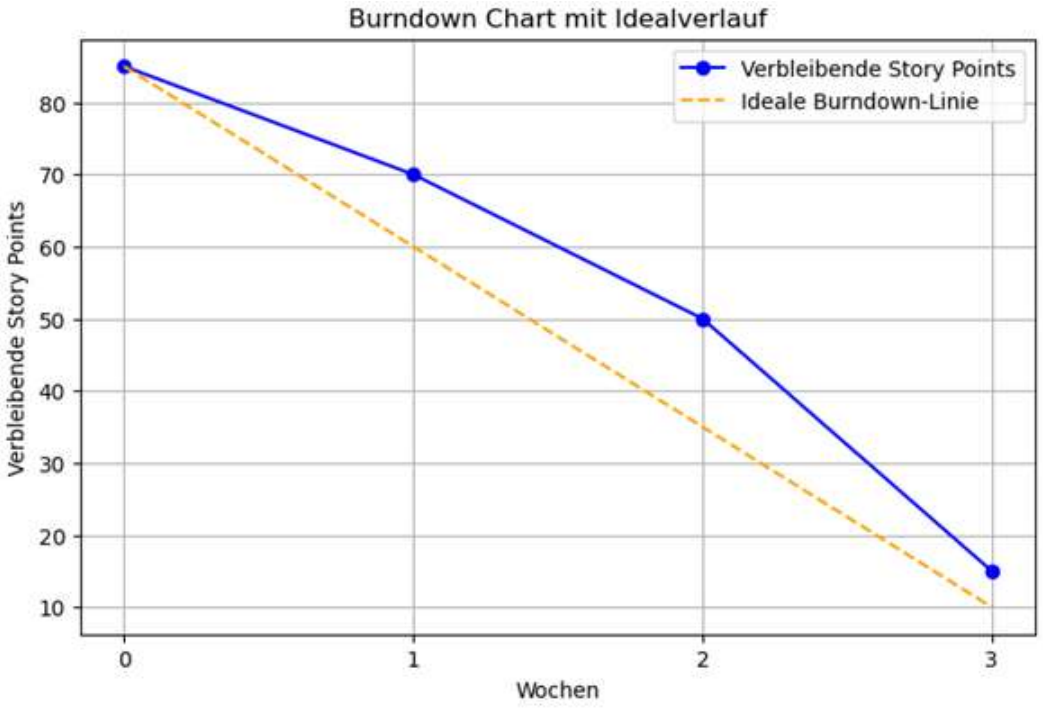
- Missverständnisse und Streitigkeiten – Endpoints, Doppelte Nachrichten etc.

### 6.2.4 Sprint Zusammenfassung

5. Zuerst haben wir uns um das Design gekümmert.
6. Wir haben die Login Page mit Funktionalität eingebaut
7. Wir haben die API-Schnittstelle erstellt.
8. Aufbau und Design der Steuerung Pages

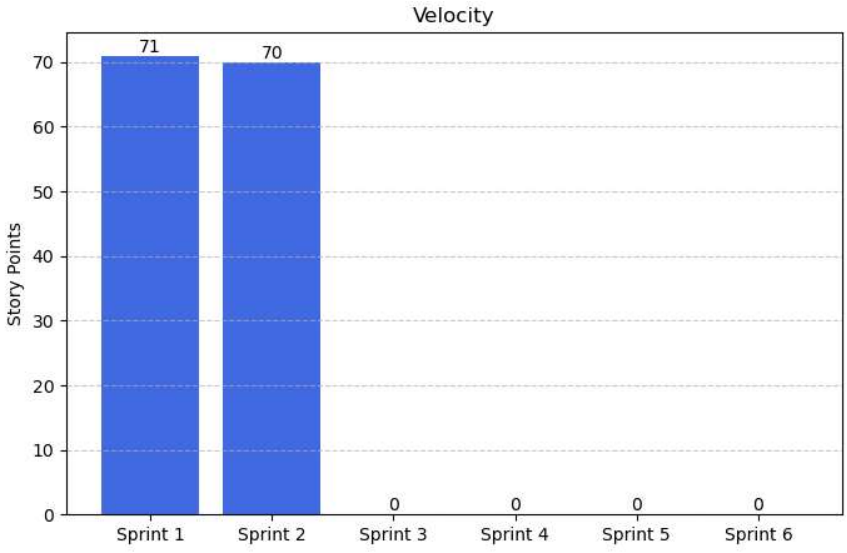
**Burndownchart:**

Geplante Storypoints: **85**
Erledigte Storypoints: **70**
Offene Storypoints: **15**
Verbleibende Storypoints in Product Backlog: **155**




## Sprint Velocity

In diesem Sprint hatten wir eine Velocity von 70. Mit einer solchen Geschwindigkeit wäre man in 2-3 Sprints fertig, also ca. 1 Sprint vor dem geplanten Ende. Auch der erste Sprint hatte eine ähnliche Velocity von 70, das zeigt, dass die Velocity gehalten werden konnte.



# 6.3 Sprint 3

## 6.3.1 Sprint Planung

Dauer 19.03.2025 – 23.04.2025

Ausgewählte User Stories:

- Kartografieren:

  Der User kann den Kartographie-Modus auswählen. Dort drückt er auf Drive und es wird begonnen, den Raum abzufahren. Am Ende kann der User den Button erneut betätigen und es öffnet sich ein PopUp in der die Map aufgezeichnet wird. Darin werden die einzelnen Punkte, aber auch die Außenlinie und Hindernisse angezeigt.

  ➔ 100 Story Points

- Route definieren und abfahren:
  Der User kann beim Automatischen Fahren eine Route definieren. Dafür muss er nur auf den Button Route definieren klicken. Dort kann er einzelne Checkpoints/Abschnitte definieren, darunter kann man die Dauer, Geschwindigkeit und Farbe in diesem Bereich speichern. Nebenbei wird die Route automatisch daneben angezeigt. Checkpoints werden ebenfalls angezeigt und können auch wieder gelöscht werden. Im nächsten Schritt kann der User diese Route speichern. Diese wird dann automatisch an das Backend gesendet. Für das Abfahren kann eine gespeicherte Route selektiert werden. Danach soll der user auf Drive klicken und die Route wird er abgefahren. Mit erneutem Klicken wird dies wieder beendet.

  ➔ 40 Story Points

- Datenvisualisierung:
  Kommunikation mit dem Backend um Detaildaten wie die Batterieladung zu holen. Geschwindigkeit etc. werden berechnet und visualisiert.
  ➔ 15 Story Points

Anzahl Story Points: 155

Ausgewählte Punkte aus der Impediment Liste:

- Verzögerung bei Kartografie. Probleme im Backend lösen Verzögerung und Wartezeiten aus.
- Kartografie generell unvollständig und schlechte Planung → viel zu große user Stories.

### 6.3.2 Sprint Demo

In diesem Sprint gab es leider Probleme, die durch einen Merge-Konflikt ausgelöst wurden. Nach einigen Stunden konnte dieses Problem behoben werden. Daher auch allgemeine Verzögerung. Das Definieren und Abfahren der Route verlief sehr gut. Auch die Daten Visualisierung konnte vervollständigt werden. Leider konnte das Kartografie-Modul nicht fertiggestellt werden, da es Probleme im Backend gab, die auch im Frontend zu Verzögerungen führten. Auch das Kartografieren wurde im Sprint noch begonnen. Es handelt sich dabei um eine sehr große User-Story (Fehler in der Planung), wodurch sie am Ende des Sprints nicht fertiggestellt werden konnte. Insgesamt wurde im Sprint 3 eine Velocity von ~ 55 erreicht, da die Kartografie-User Story mit 100 Story Points nicht fertiggestellt wurde. Diese wurde wieder in den Backlog zurückgelegt.

### 6.3.3 Sprint Retrospektive

Besonders gut verlief die Kommunikation mit dem Backend, mit dem wir ständig im Austausch waren. In diesem Sprint wurde das automatische Abfahren, Auswählen und Definieren von Routen vollständig umgesetzt. Des Weiteren haben wir im Team gute Zusammenarbeit genossen, so haben wir oft gemeinsam Probleme und Fehler gelöst. Zu Beginn hatten wir aber mit einem Fehler beim Mergen zu tun, der die gesamte Entwicklung des Frontend gefährdete und verzögerte. Da wir leider in der Planung schon den Fehler gemacht haben, die Kartografie-user Story zu groß zu machen, könnte diese niemals in einem Sprint fertiggestellt werden können, wodurch diese in diesem Sprint trotz Verzögerung, ausgelöst durch das Backend, begonnen wurde. Am Ende haben wir in diesem Sprint nur eine Velocity von 55 erreicht. Jedoch wurde auch festgestellt, dass das Styling überabreitet, werden muss und ein Logout-Button als Feature noch fehlen.

Verbesserungsbedarf gibt es bei der Aufgabenverteilung und bei dem Vorausplanen.

Auflistung der Impediment Taskliste:

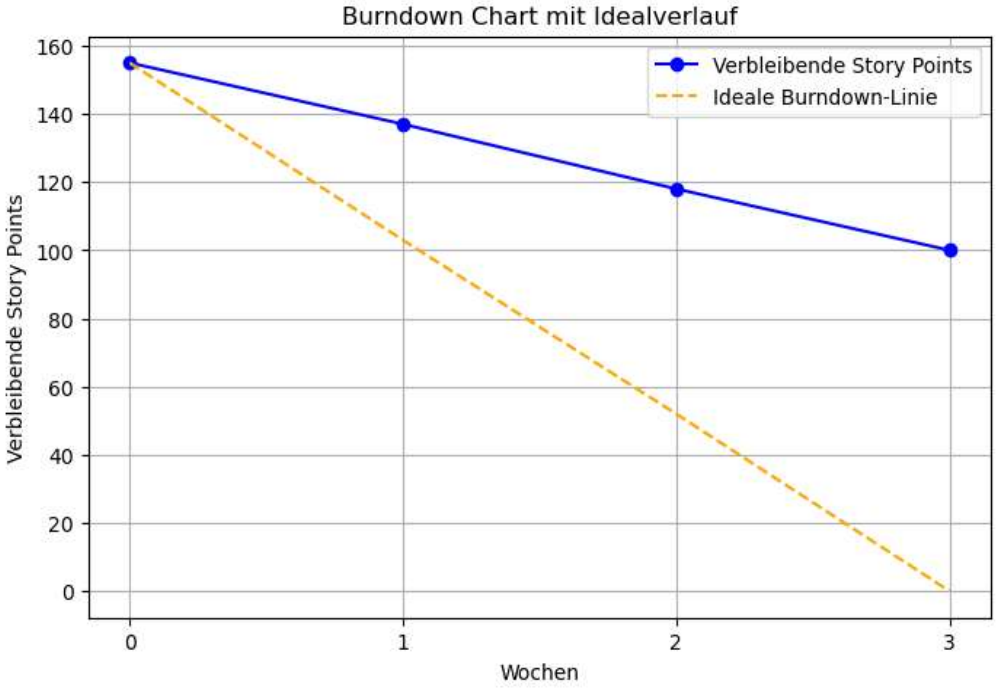
- Schlechte Planung
- Bessere Kommunikation
- Mehr Vorsicht bei Git-Interaktionen

### 6.3.4 Sprint Zusammenfassung

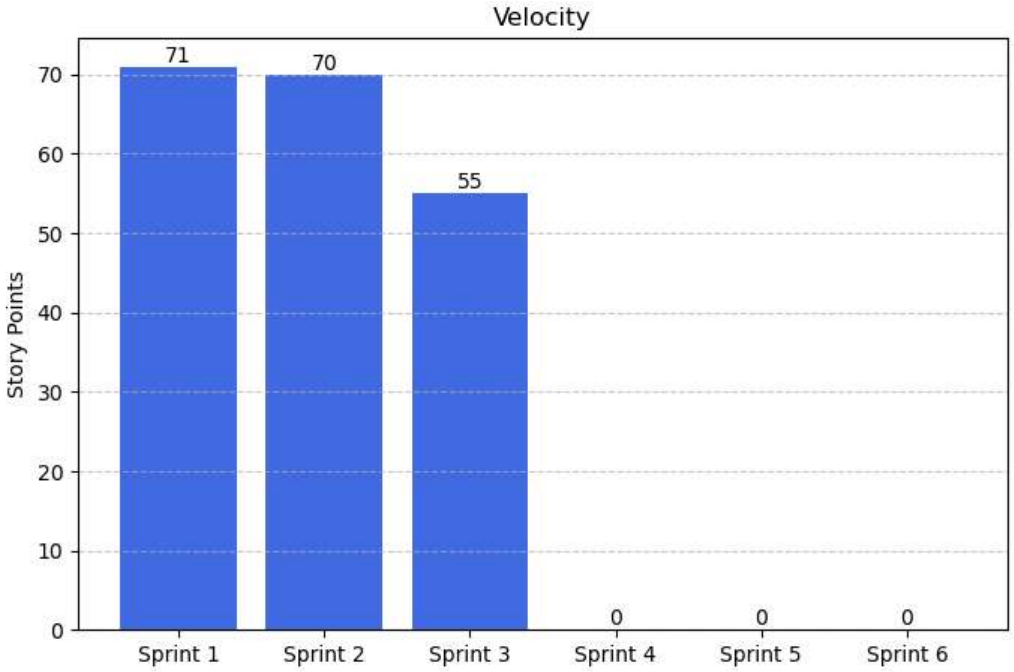
Zuerst haben wir uns um das Design gekümmert.
Wir haben die Login Page mit Funktionalität eingebaut
Wir haben die API-Schnittstelle erstellt.
Aufbau und Design der Steuerung Pages

**Burndownchart:**

Geplante Storypoints: **155**
Erledigte Storypoints: **55**
Offene Storypoints: **100**
Verbleibende Storypoints in Product Backlog: **115**




**Sprint Velocity:** In diesem Sprint hatten wir eine Velocity von 55. Mit einer solchen Geschwindigkeit wäre man in 1-2 Sprints fertig. Da wir uns für diesen Sprint auch schon die Map mit 100 Story Points vorgenommen haben, konnte dies leider nicht fertig gestellt werden.



# 6.4 Sprint 4

## 6.4.1 Sprint Planung

Dauer 24.04.2025 – 13.05.2025

Ausgewählte User Stories:

- Kartografieren:

  Der User kann den Kartographie-Modus auswählen. Dort drückt er auf Drive und es wird begonnen, den Raum abzufahren. Am Ende kann der User den Button erneut betätigen und es öffnet sich ein PopUp in der die Map aufgezeichnet wird. Darin werden die einzelnen Punkte, aber auch die Außenlinie und Hindernisse angezeigt. Fortsetzung der User Story vom vorherigen Sprint.

  ➔ 100 Story Points

  ➔ Logout Button
  Der User kann die Verbindung mit dem MBot wieder trennen, indem er er den dafür erstellten Button betätigt. Dabei kommt man wieder zur Landing-Page zurück.

  ➔ 20 Story Points

- Reworked Mobile UI
  Die UI vom Frontend in der mobilen Ansicht verbessern und verschönern.
  ➔ 15 Story Points

Anzahl Story Points: 135

Ausgewählte Punkte aus der Impediment Liste:

- Verzögerung bei Kartografie. Probleme im Backend lösen Verzögerung und Wartezeiten aus.
- Kartografie generell unvollständig und schlechte Planung → viel zu große user Stories.

In diesem Sprint wurden zwei weitere User-Stories, der Logout-Button (20) und Design-Rework (15) dem Backlog, ausgehend von der Retrospektive von Sprint 3 hinzugefügt.

### 6.4.2 Sprint Demo

In diesem Sprint gab es leider Probleme, vor allem bei dem Kartografieren. Zuerst gab es Rückschläge im Backend Team, wodurch auch im Frontend das Kartografieren verzögert wurde. Diese Verzögerung zieht sich durch den ganzen Sprint. Nach einigen Stunden konnte dieses Problem behoben werden. Daher auch allgemeine Verzögerung. Leider konnte das Kartografie-Modul nicht fertiggestellt werden, da es Probleme im Backend gab, die auch im Frontend zu Verzögerungen führten. Das Kartografieren wurde fortgesetzt, da es sich um eine sehr große User-Story (Fehler in der Planung) handelt, wodurch sie am Ende des Sprints nicht fertiggestellt werden konnte. Insgesamt erreichten wurde daher nur eine Velocity von ~ 0 erreicht, da die Kartografie-User Story mit 100 Story Points, auch in diesem Sprint, nicht fertiggestellt werden konnte. Deren Codierung selbst ist bereits fertig, es fehlen aber noch das Testen mit Unit-Tests und die Dokumentation, wodurch es der Definition of Done nicht entspricht.

### 6.4.3 Sprint Retrospektive

Besonders gut verlief die Kommunikation mit dem Backend, mit dem wir trotz Verzögerung ständig im Austausch waren. In diesem Sprint haben wir im Team gute Zusammenarbeit genossen, so haben wir oft gemeinsam Probleme und Fehler gelöst. Da wir leider in der Planung schon den Fehler gemacht haben, die Kartografie-User Story zu groß zu machen, könnte diese niemals in einem Sprint fertiggestellt werden können, wodurch diese in diesem Sprint trotz Verzögerung, ausgelöst durch das Backend, fortgesetzt wurde. Leider konnte diese Userstory. Am Ende haben wir in diesem Sprint nur eine Velocity von 0 erreicht.

Verbesserungsbedarf gibt es bei der Aufgabenverteilung und bei dem Vorausplanen.

Auflistung der Impediment Taskliste:

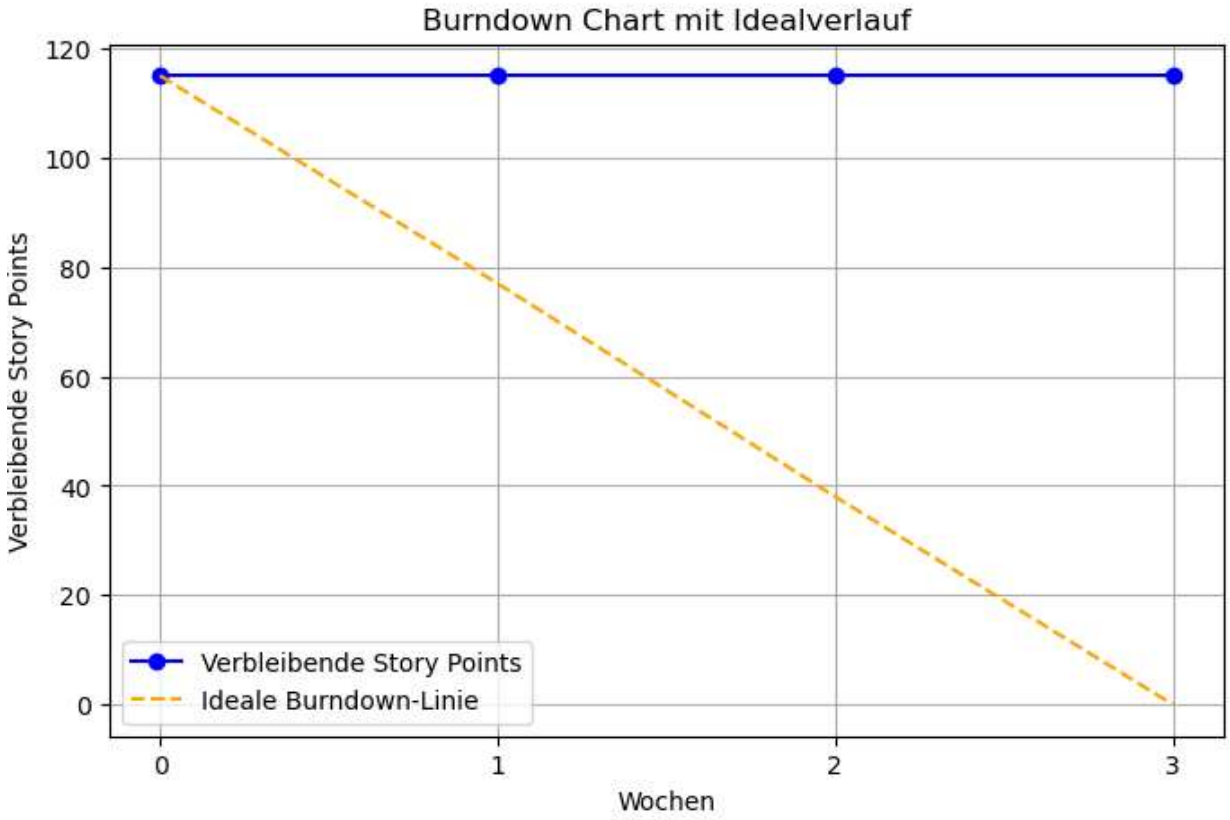
- Schlechte Planung
- Bessere Kommunikation
- Mehr Vorsicht bei Git-Interaktionen

### 6.4.4 Sprint Zusammenfassung

In diesem Sprint wurde sich erneut mit dem Design befasst und es überarbeitet, außerdem hat er sich um die vollständige Implementation der Mobile Ansicht bemüht.
Des Weiteren wurde in diesem Sprint das Kartografieren fast fertiggestellt. Durch anfängliche Verzögerungen mit dem Backend auch in diesem Sprint konnte es schlussendlich nicht entsprechend der Defintion of Done erstellt werden.
Außerdem wurden weitere Verbesserungen und die Implementierung eines Logout-Buttons begonnen. Die jeweiligen User Stories für den Logout-Button und das UI Rework wurden dem Backlog hinzugefügt.

## Burndownchart:

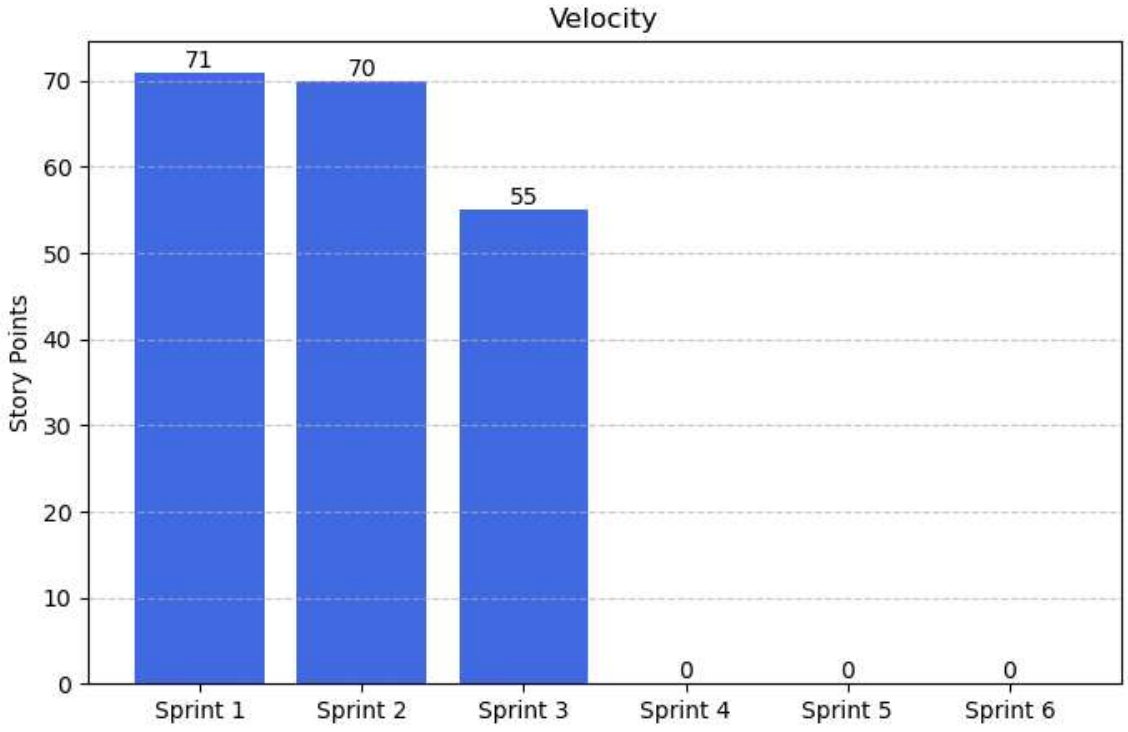
Geplante Storypoints: **115**
Erledigte Storypoints: **0**
Offene Storypoints: **115**




## Sprint Velocity

In diesem Sprint hatten wir eine Velocity von 0. Dies ist er Tatsache geschuldet, dass keine User Story entsprechend der Definition of Done fertiggestellt werden konnte.

**Velocity Sprint 4:** **0**
**Velocity Durchschnitt:** **65,33**


Velocity
Story Points
0
10
20
30
40
50
60
70
71
70
55
0
0
0
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5
Sprint 6

# 6.5 Sprint 5:

## 6.5.1 Sprint Planung

Dauer 14.05.2025 – 10.06.2025

Ausgewählte User Stories:

- Kartografieren:

  Der User kann den Kartographie-Modus auswählen. Dort drückt er auf Drive und es wird begonnen, den Raum abzufahren. Am Ende kann der User den Button erneut betätigen und es öffnet sich ein PopUp in der die Map aufgezeichnet wird. Darin werden die einzelnen Punkte, aber auch die Außenlinie und Hindernisse angezeigt.

  Diese Userstory wurde bereits in den vorherigen Sprints umfangreich bearbeitet und beinahe fertiggestellt. Diese Userstory wurde nun noch entsprechend der Definition of Done finalisiert.

  ➔ 100 Story Points

- Logout Button

  Der User kann die Verbindung mit dem MBot wieder trennen, indem er den dafür erstellten Button betätigt. Dabei kommt man wieder zur Landing-Page zurück. Auch das wurde fertiggestellt. Diese User Story wurde ja im Sprint 3 hinzugefügt.

  ➔ 20 Story Points

- Reworked Mobile UI

  Die UI vom Frontend in der mobilen Ansicht verbessern und verschönern. Diese User Story wurde aus dem 4. Sprint fortgesetzt und nun ebenfalls fertiggestellt.

  ➔ 15 Story Points

- Dokumentation fertigstellen

  In diesem Sprint wird die Dokumentation fertiggestellt und etwaige fehlende Unittests nachgeholt. Außerdem wurde die Abschlusspräsentation erstellt.

  ➔ 20 Story Points

Anzahl Story Points: 155

### 6.5.2 Sprint Demo

In diesem Sprint gab es keine Probleme mehr. Diese wurden alle im Zuge vorheriger Sprints schon gelöst. Auch wenn es in diesem Sprints Verzögerungen gab, konnte in diesem Sprint alles problemlos fertiggestellt werden ohne Verzögerungen und Schwierigkeiten. Schlussendlich konnte das Projekt pünktlich fertiggestellt werden.

### 6.5.3 Sprint Retrospektive

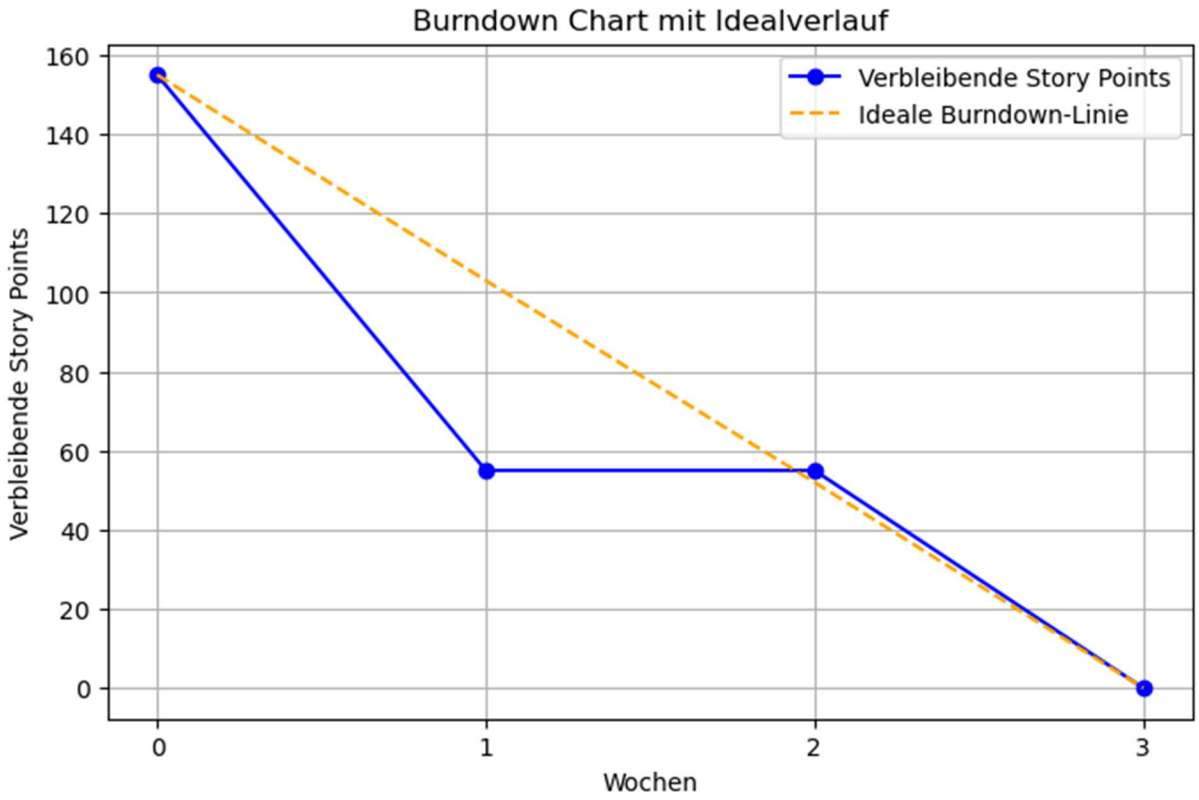
Besonders gut verlief die Kommunikation mit dem Backend-Team. Außerdem hatten wir in diesem Sprint kaum noch Probleme, da alle laufenden User Stories bis auf die Dokumentation bereits davor ausgiebig bearbeitet wurden. In diesem Sprint haben wir im Team gute Zusammenarbeit genossen. Trotz Fehler in der Planung, die sich in den vorherigen Sprints herausgestellt haben konnten, schlussendlich alle Users Stories pünktlich zum Projektende fertiggestellt werden. Am Ende haben wir in diesem Sprint eine Velocity von 155 erreicht. Dies ist der Tatsache geschuldet, das im Sprint 4 nichts vollständig fertiggestellt werden konnte.

### 6.5.4 Sprint Zusammenfassung

Im letzten Sprint wurden alle offenen Aufgaben erfolgreich abgeschlossen. Der Kartografierungsmodus wurde finalisiert und erfüllt nun vollständig die Anforderungen inklusive Anzeige von Punkten, Außenlinie und Hindernissen. Zudem wurde ein Logout-Button implementiert, der die Verbindung zum MBot trennt und den Nutzer zurück zur Landing-Page führt. Die mobile Benutzeroberfläche wurde überarbeitet und verbessert. Auch die Projektdokumentation und die Abschlusspräsentation wurden fertiggestellt. Insgesamt wurden 155 Story Points erreicht. Es gab keine größeren Probleme – die gute Vorbereitung aus den vorherigen Sprints sowie eine reibungslose Zusammenarbeit, insbesondere mit dem Backend, ermöglichten einen pünktlichen und erfolgreichen Projektabschluss.

**Burndownchart:**

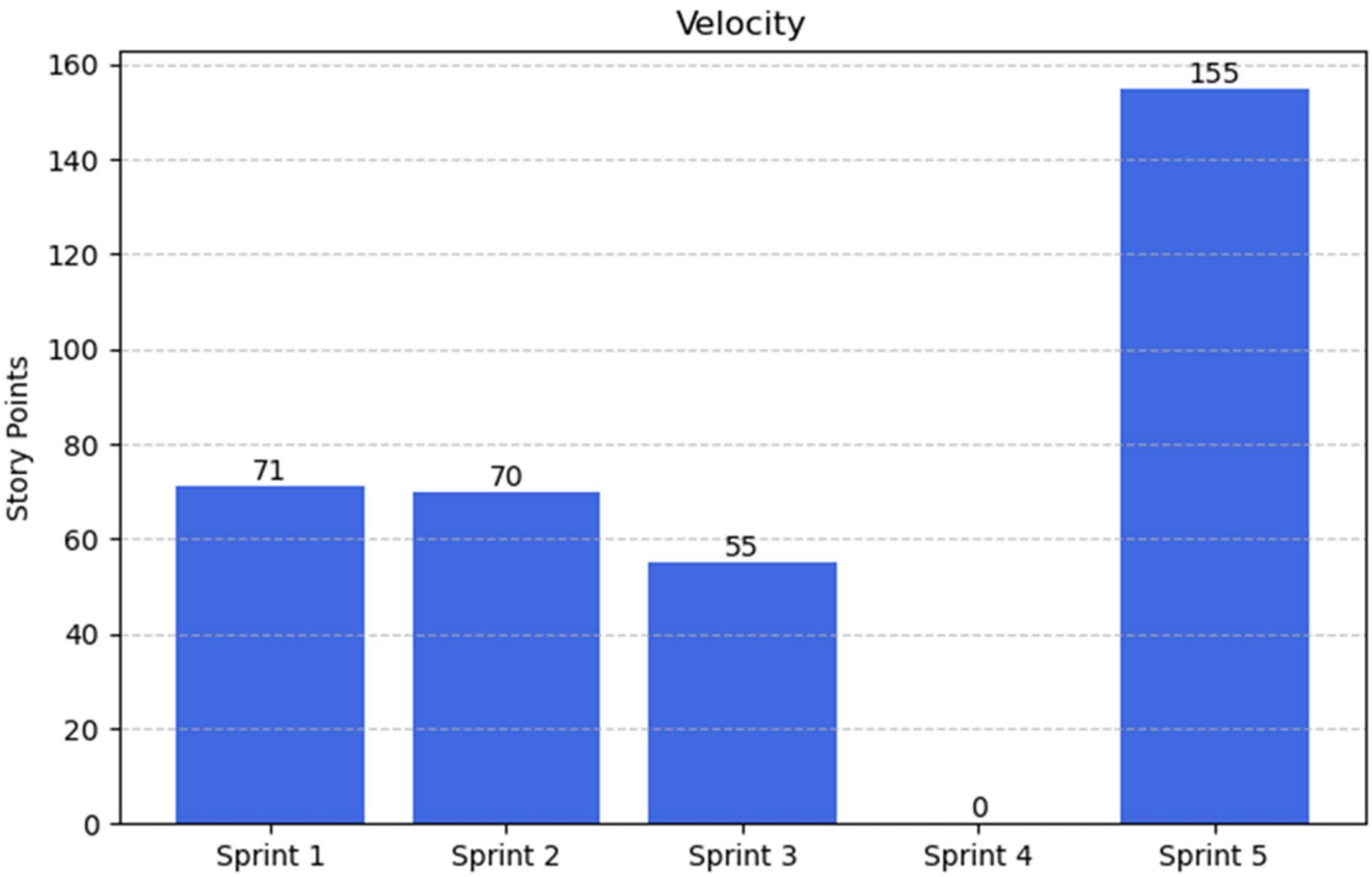
Geplante Storypoints: **155**
Erledigte Storypoints: **155**
Offene Storypoints: **0**



## Sprint Velocity

In diesem Sprint hatten wir eine Velocity von 155. Wir konnten die schon im Sprint 4 weit fortgeschrittenen User Stories fertigstellen und das Projekt erfolgreich finalisieren.

**Velocity Sprint 5:** **155**
**Velocity Durchschnitt:** **71**



# 7. Installation / Software deployment

Um das Projekt lokal auszuführen sind folgende Schritte notwendig:

Zunächst müssen alle erforderlichen Abhängigkeiten installiert werden. Dies geschieht durch den Befehl *npm install*, der die in der package.json definierten Bibliotheken und Tools (wie React, Webpack oder Babel) herunterlädt und im Projektverzeichnis unter node_modules ablegt. Dieser Schritt ist essenziell, da erst dadurch die notwendigen Funktionen für die Entwicklung und den Betrieb der Anwendung verfügbar werden.
Anschließend kann die Anwendung im Entwicklungsmodus gestartet werden. Der Befehl *npm run start* initialisiert einen lokalen Entwicklungsserver (typischerweise auf Port 3000) und öffnet die Anwendung automatisch im Standard-Browser. Dieser Server unterstützt **Hot-Reloading**, was bedeutet, dass Code-Änderungen sofort angezeigt werden, ohne dass die Anwendung manuell neu gestartet werden muss. Dies beschleunigt die Entwicklung erheblich.
Für die Bereitstellung in einer Produktionsumgebung wird der Befehl npm run build verwendet. Dieser erzeugt eine optimierte Version der Anwendung, bei der der Code komprimiert und für eine effiziente Auslieferung vorbereitet wird. Die generierten Dateien werden im Ordner build abgelegt und können auf einen Web-Server hochgeladen werden.

# 8. Projektabschluss

## 8.1 Projektzusammenfassung

Das Projekt konnte trotz einiger Herausforderungen erfolgreich abgeschlossen werden. Besonders die teilweise Abhängigkeit vom Backend führte zu Verzögerungen, insbesondere bei der Kartografiefunktion, die aufgrund ihrer Komplexität über mehrere Sprints hinweg bearbeitet werden musste. Dennoch gelang es dem Team, durch iterative Verbesserungen und enge Zusammenarbeit alle geplanten Features umzusetzen.

### Wichtige Meilensteine:

- Sprint 1 (71 Story Points):

  Grundlegende Funktionalitäten wie das Mockup-Design, die API-Anbindung und die Steuerungs-UI wurden erfolgreich implementiert. Die Kommunikation mit dem Backend verlief reibungslos.
- Sprint 2 (70 Story Points):

  Die manuelle Steuerung und Routenspeicherung wurden fertiggestellt, während die Datenvisualisierung aufgrund von Backend-Verzögerungen verschoben werden musste.

- Sprint 3 (55 Story Points):

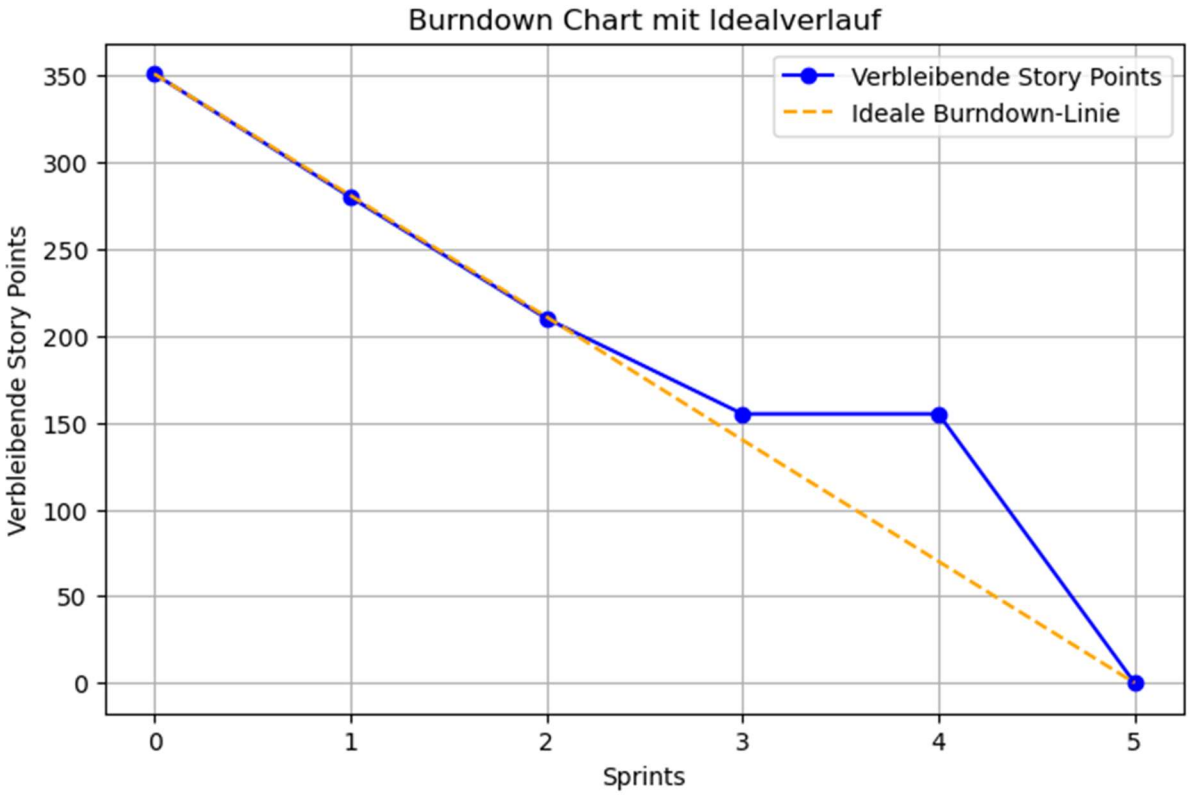
  Das automatische Routenmanagement und die Datenvisualisierung wurden abgeschlossen, doch die Kartografiefunktion erwies sich als zu groß für einen Sprint. Ein Merge-Konflikt kostete zusätzlich Zeit.

- Sprint 4 (0 Story Points):

  Aufgrund anhaltender Backend-Probleme konnte keine User Story finalisiert werden, obwohl die Entwicklung weit fortgeschritten war.

- Sprint 5 (155 Story Points):

  Alle offenen Aufgaben – Kartografie, Logout-Button, Mobile-UI-Optimierung und Dokumentation – wurden pünktlich fertiggestellt.



### Erkenntnisse & Optimierungsbedarf:

Planung:

Die Aufteilung zu großer User Stories (z. B. Kartografie) führte zu unvollständigen Sprints. Künftig sollten solche Features in kleinere, besser schätzbare Tasks unterteilt werden.

Kommunikation:

Der enge Austausch mit dem Backend-Team war entscheidend, jedoch gab es zeitweise Verzögerungen durch unklare Schnittstellen oder technische Probleme.

### Fazit:

Trotz der Herausforderungen konnte das Projekt innerhalb des geplanten Zeitraums abgeschlossen werden. Die stabile Velocity (Durchschnitt: ~70.2 Story Points/Sprint) und die erfolgreiche Umsetzung aller Kernfeatures zeigen die Effektivität des Teams. Für zukünftige Projekte sollten besonders die Backend-Abhängigkeiten früher berücksichtigt und User Stories kleinteiliger geplant werden.

Abschließend erwies sich die agile Vorgehensweise als erfolgreich, da sie Flexibilität für Anpassungen bot und eine kontinuierliche Verbesserung ermöglichte.

## 8.2 Attachments

- Mockup – Visualisiert das grobe/geplante Design
- Projektdatei als ZIP-File mit dem Quellcode
- UML-Diagramme zur Darstellung der Softwarearchitektur und des technischen Designs
- Diagramme, Burndown-Charts zur Visualisierung des Fortschritts
- Definition of Done und Ziele und Anforderungen erarbeitet + Mockup
- Review & - Abschlusspräsentation

Alle Anlagen sind auch im Projektordner zu finden.